An exploration of quantitative models and algorithms for vehicle routing optimization and traveling salesman problems

Oskari Lähdeaho\textsuperscript{a,*,} Olli-Pekka Hilmola\textsuperscript{a,b,c}

\textsuperscript{a} LUT University, Kauppakatu 13, 45100 Kouvola, Finland
\textsuperscript{b} Tallinn University of Technology (Taltech), Estonian Maritime Academy, Kopli 101, 11712 Tallinn, Estonia
\textsuperscript{c} Department of Industrial Engineering and Management, University of Gävle, SE-801 76 Gävle, Sweden

**ARTICLE INFO**

**Keywords:**
Vehicle Route Optimization
Traveling Salesman Problem
Branch and Bound
Logistics
Parallel Computing
Graphics Processing Unit

**ABSTRACT**

This study presents optimization models for large vehicle routing problems using a spreadsheet solver and Python programming language with extended graphic card boosting computing power. Near optimality is feasible and attainable with spreadsheet tools and models for solving real-life problems. However, increasing the availability of additional computing power through graphics processing and visualization is now a viable option for decision-makers and problem-solvers. This study shows that decision-makers can solve vehicle routing optimization problems with limited access to high-end optimization tools. This study shows managers and decision-makers can use vehicle routing optimization even with limited access to sophisticated optimization tools.

1. Introduction

Optimization algorithms providers usually promise to solve any real-world problem (e.g., in manufacturing or transportation) in a matter of seconds. However, often the optimization problem proves to be more complex than at the first glance. It is so that real world problems require attention to details and general solutions are not viable. Therefore, it is important to educate future professionals on how to construct algorithms for solving problems such as vehicle route optimization, for them to later utilize this knowledge in a unique setting of a specific company. Even seemingly straightforward algorithm-based optimization problems in real-world business management require modifications to the existing algorithms, multiple iterations of developing that algorithm to reach desired solutions as well as educated personnel to carry out this process [1].

In logistics industries, vehicle routing optimization is vital for running competitive operations since optimized routes and timing decrease operation costs and increase the quality of the service [2]. Multimodal transportation can also benefit from such optimization [3]. Sustainability and environmental friendliness of logistics operations is a large concern in modern industries, and implementing new practices and technologies efficiently requires inclusion of them to the optimization practices [4].

This article explores optimization problems through master’s level university education in a degree related to innovation and logistics. The authors carried out a course on quantitative models and methods in logistics, which is described more in detail in the latter parts of the article. The focus of this article is also on approaches and tools that are accessible to smaller companies. In other words, optimization tools that are openly available and do not require extensive investments in computer software or hardware. Optimization problems such as Travelling Salesman Problem (TSP) has been used in education, as it offers an effective way for students to learn principles in optimization and quantitative methods [5,6]. These are skills often desired by companies operating in logistics industries, when hiring young talent. In this research work we introduce two different modelling and optimization techniques – namely spreadsheet (solver) based and a Python programming language developed, where graphics processing unit’s (GPU) computing power is being utilized. In addition, a Python program that does not use GPU is presented. TSP itself is an interesting problem that has attracted many solvers and different approaches from different backgrounds and disciplines. While the problem and proposed solutions have existed for decades, new ways to solve the problem more efficiently are constantly found. There are even competitions held on solving TSP problems (such as [7–9]). Research problem in this work could be stated as follows: How to model and solve in the university course larger TSP
problem? What kind of skills and software-hardware combination is needed to build and solve these models?

The rest of this article is structured as follows. Literature review related to vehicle routing optimization, used algorithms and education related to them is presented in the second section. Thereafter, the third section describes our approach for optimization algorithm education. The following section combines the different algorithms and their results used to approach the chosen optimization problem. Finally, the article is concluded with discussions reflecting the existing literature with the findings of this research work.

2. Literature review

Vehicle Routing Problem solving is an established task both in academia and practice. There is a myriad of different highly sophisticated approaches and models (e.g., Genetic Algorithms [10]; Neural Networks [11]; Bee Colony Algorithms [12]; Parallel algorithms [13]). However, solutions such as spreadsheet software-based optimization are also quite effective in many cases [14]. To prove the efficiency of vehicle routing optimization, real-life observations are gathered from the optimized routes [11,12]. In addition, vehicle route optimization benefits from observations related to the surroundings and setting, such as other vehicles, other traffic, time of the day, and date [15]. It is difficult to predict effects of unique regional and cultural characteristics on traffic and route optimization without experimentation [16]. Companies can improve their logistics operations with this kind of data [17]. Especially, when the operations take place in busy areas, such as cities [18,19]. In addition, last-mile delivery related optimization requires real-life data as these deliveries are rarely uniform and related factors are difficult to predict [20]. Sufficiently gathered real-life data could also be used to create dynamic vehicle routing optimization, as well as to other logistics operations improvements such as tracing and tracking within a supply network [21]. These optimization problem solutions are also desired by smaller actors, who might not have the needed resources to run sophisticated models, and here the open-source solutions might be more appropriate [2].

There are many vehicle routing optimization challenges in traditional road transportation logistics that require further research, such as cargo capacity limits, time window limits, pickups in addition to deliveries, differing fleet characteristics, multiple depots and collaborations between companies, split deliveries in one cargo unit, and truck and trailer with separate delivery locations, to name a few [22-25].

There has been proposed vehicle routing optimization models that aim to collaboration between different logistics actors to efficiently utilize available cargo capacity [26,27]. Last-mile deliveries are increasingly important in the contemporary world where e-commerce is still growing, and route optimization can be used to improve this part of transportation activities and related decision-making [25]. This part of
Supply chains is also highly competed between service providers, which creates a need for efficient methods in route optimization [28]. Customer absence provides further challenges in establishing last-mile deliveries [29]. Vehicle routing optimization is not tied to only road transportation problems, since they can open avenues in multimodal transportation, e.g., last-mile aerial drone deliveries [3,30].

Vehicle routing optimization has been increasingly associated also with environmental sustainability in recent two decades [31]. Implementation of environmental sustainability practices in logistics with electric vehicles, their charging, reduced energy consumption, and emissions mitigation can be carried out more fluently with optimized vehicle routes and timings [4]. Many instances of environmental sustainability approach in Vehicle Route Optimization can be found, where fuel consumption in transportation is used as factor for produced emissions [32]. Vehicle routing optimization can also enable other energy efficiency related efforts, e.g., oil and gas recovery in refined oil distribution [33]. In addition to conventional vehicles, route optimization with electric vehicles and their need to visit charging stations has been carried out [23]. More detailed models also exist, which consider vehicle technology, average speed, vehicle mileage, engine temperature, road height, gross weight, road gradient, ambient temperature, and ambient humidity to produce more accurate optimization [34]. When optimizing with multiple factors, the approach on which factors are preferred over others becomes important (e.g., balanced use of capacity over the fastest possible route [35]). Multi-objective models are created to optimize between multiple such factors e.g., in waste management [36] or medical waste management [37]. In similar manner, some models consider economic and social sustainability in addition to environmental sustainability in vehicle routing optimization [38,39].

TSP is used in education as it provides a way to learn programming and optimization with a graspable setting (that being physical objects moving within realistic time and space). For example, game-based education with TSP has been used to achieve these educational goals [5]. Because optimization problems are usually quite difficult and require existing knowledge on quantitative models, some have built tools to provide such problems for educational purposes that are easier to grasp for the students [6]. Setting up the course as a project with a real-world optimization problem to solve improves student involvement and efficiency of learning [40]. Establishing the optimization problem as a game can also increase student learning of logistics and transportation and required related skills to carry out optimization [41].

3. Approach to education with optimization algorithms

TSP concerns setting, where salesman needs to visit n locations (e.g., cities or towns), and in the end of this travel activity will return to the starting place (first location). This travel route, or actually a tour, should be the shortest possible. Even if this sounds like an easy problem to solve, it is NP-hard. TSP could be formulated as Linear Integer Programming problem as follows (e.g., [42–44]):

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{Subject to}
\]

![Fig. 3. TSP optimization problem setting, 51 cities in China [47].](image)
\[
\sum_{i=1}^{n} x_{ij} = 1, \ldots, n, \quad (2)
\]

\[
\sum_{j=1}^{n} x_{ij} = 1, \ldots, n, \quad (3)
\]

\[
x_{ij} = 0 \text{ or } 1 \quad (4)
\]

where \(c_{ij}\) is the cost (or distance) of traveling from city \(i\) to \(j\), and \(x_{ij}\) is binary selection whether this travel from \(i\) to \(j\) is being selected.

With the above objective function and its constraints, it is typically so that minimized solutions produce answers, which contain number of subtours, and minimized tour is not only one tour, but numerous tours instead. This is not the correct answer for TSP, and therefore additional constraints are typically being added to eliminate subtours [42,44]. For larger problems it is difficult to find an exact solution (or lowest possible) cost (or distance) tour. Therefore, algorithms have been developed, which will produce solutions being rather near the optimal solution. Most well-known software for such is Concorde, which has tried to gather all known (with new ones) TSP algorithms together [42]. Another one is Lin-Kernighan Heuristics [43]. In this research work we introduce two alternative approaches, that of using spreadsheet solver algorithm and another one based on graphics card and Python programming language. Additionally, we present a Python algorithm that can produce optimization results in reasonable time without using GPU.

The experimental area of this research takes place in a course on quantitative models and methods of logistics, which was carried out by the authors in the spring of 2023. The course is a part of the curriculum for master’s degree students in innovation and logistics program. As the course is optional for everyone enrolled in the program, making it interesting for the students becomes more relevant (in order to attract enough students to run proper course). Our approach was to organize a seminar day, where the students present their solution to an optimization problem of their choosing. These seminar works were based on problems presented throughout the course, including classics such as the TSP. TSP refers to an optimization problem where there is (in simple form) a travelling salesman who must travel through number of cities, and the solver must find out the shortest route that travels through all those cities. Please see Fig. 1 for a simple illustration of the problem.

Some basic solutions to various problems were given in the course, but the catch was to solve more complex problems, and the students could compete on who comes up with the “best” solution. This approach was chosen to amplify students’ learning through engagement and competitive setting. While the seminar day contained interesting solutions on various optimization problems in logistics and supply chain management, this article focuses on the TSP.

During the course lectures, students were familiarized with TSP presenting a simple problem of 5–10 locations and simple brute force solution algorithm. This refers to a solution where every possible route is calculated and, in the end, the shortest one is found. Before this in lectures it was also introduced, how TSP problem may be solved using linear integer programming by adding numerous constraints to achieve...
optimal route (and possibly applying branch and bound; like introduced in [45]). To give the students some other approaches for TSP, the branch and bound approach was presented. This refers to an approach, where the problem is divided into multiple branches. These branches are gradually compared, and longer routes are eliminating during the process. In other words, the problem is divided into smaller parts, which should ease the required computing power to reach an optimum solution. Please see Fig. 2 for a simple illustration for this method. This illustration assumes that the problem is same as in Fig. 1, and the allocation of the nodes is accurate depending on their distance from each other (e.g., Node 1 is near nodes 2, 3, and 5, and it does not make sense to go to node 4 from 1). Of course, different problems require different solutions, and in many cases, branch and bound approach might not be the best or fastest way to reach an optimum solution.

However, as the seminar work must present a more sophisticated solution to a more difficult optimization problem, 51 cities in China were chosen according to their population, geographical location, and relevance in national and global supply chains. The studied cities are same as in [46]. As is the case in almost everywhere, urbanization is a trend in China. Thus, the need for supply chains routes between the largest cities can be seen only growing. Depending on the calculation method, there is already 200–300 cities with over million inhabitants in China. The cities studied in this work are Shanghai, Beijing, Guangzhou, Tianjin, Chongqing, Chengdu, Shenzhen, Harbin, Wuhan, Hangzhou, Xi’an, Shenyang, Dongguan, Nanjing, Foshan, Jinan, Wenzhou, Qingdao, Quanzhou, Shantou, Changsha, Suzhou, Zhengzhou, Dalian, Kunming, Wuxi, Xiamen, Changchun, Ningbo, Nanning, Taiyuan, Hefei, Changzhou, Tangshan, Zhongshan, Xuzhou, Guiyang, Ürümqi, Zibo, Shijiazhuang, Lianyungang, Fuzhou, Huai’an, Linyi, Lanzhou, Yangzhou, Nanchang, Huizhou, Nantong, Xiangyang, and Zaozhuang. This problem setting is illustrated in Fig. 3. Indeed, TSP with 51 cities cannot be solved with simple brute force algorithm in a reasonable time. The problem could be also made more difficult by adding multiple travelling salesmen, cargo capacities, fuel usage, or time windows for deliveries, but this time we chose a large set of cities to visit. In addition, such a large problem gives possibilities for competition in terms of solving time and the optimized route gained as the solution. In smaller problems, the proposed algorithms usually reach the optimum route, but here the solutions may vary.

4. Optimization algorithm results

At first, before it was possible to develop optimization algorithms for the set problem, it was necessary to practice data gathering. The students would start with calculating distances between sets of around ten cities. Thereafter distance matrices would be formed. Multiple different optimization approaches were tried with these initial distance matrices, and after that, these different approaches were tried with larger distance matrices. The approaches ranged from simpler branch and bound approach implemented in Python programming to more sophisticated ones with parallel computing. In addition, Excel with Solver add-on provided some nice results (like introduced in [48]). Please see Figs. 4, 5 and 6 for the TSP optimization algorithms in Python and Excel. Namely Evolutionary Engine within the add-on gave competent results. As the given optimization problem is quite difficult with 51 cities, it became obvious early that computing time became an important factor. When the optimization algorithms were allowed to run for around 10 min or more, the results became better than with a shorter run time.

Since the competition for the best solution was intense, the authors wanted to see if they could come up with the shortest route for this problem. The other author worked with the Excel Solver add-on and the other with a Python algorithm that uses graphic card capabilities to run parallel computing (somewhat similarly as in [13]) to find the optimum route. In addition, a Python algorithm relying on Google OR-Tools libraries was tested. Gradually after multiple tries and comparisons between the optimizing algorithm parameters, run times, and results, both Python algorithms provided similar, slightly shorter route. The used Python script can be found in Appendix. While the Excel Solver add-on provided a route with a length of 18628.7 kilometers, the discussed Python algorithm gave a route 18610.1 kilometers long. This route is provided some nice results (like introduced in [48]). Please see Figs. 4, 5 and 6 for the TSP optimization algorithms in Python and Excel.

Fig. 5. A screenshot of Excel Solver add-on solving the TSP.
Zaozhuang, Linyi, Lianyungang, Huai’an, Xuzhou, Zhengzhou, Xiangyang, Wuhan, Changsha, Nanchang, Hefei, Nanjing, Yangzhou, Changzhou, Wuxi, Suzhou, and lastly, Nantong.

The GPU used in this work for running the algorithm with parallel computing is GeForce RTX 3080 from Nvidia. These modern and powerful graphic cards are excellent for carrying out tasks where it is necessary to repeat massive amounts of simple function calculations (as is the case with drawing graphics on monitor) due to these graphic processing units having thousands of processor cores (in comparison to only a few in Central Processing Unit or CPU). Solving TSP falls under such task, so it makes sense to use GPU for solving it. While GPUs have been used for processing data in industry settings during the last couple of decades, this approach is not very common in academic research (of course there exists some studies, such as [49]). Indeed, these modern GPUs are so sought after (e.g., by financial institutions for computing data) that consumers might find it difficult to obtain one due to constant stockouts for consumer outlets and prices being over a thousand dollars.

The algorithm for the presented optimization problem was constructed with using Nvidia cuOpt, which allows the use of GPU-acceleration for Operations Research related optimization, such as vehicle routing (Please, read more here [50]).

There are also possibilities to use Python programming for fast results in route optimization without high end GPU capabilities with Google OR-Tools libraries [51]. In the given optimization problem, this type of algorithm gave competent results in two seconds. Even after letting the algorithm search for better results over several minutes, a better route was not found. Table 1 and Fig. 8 illustrate the optimization results of this algorithm. This result shows that with even modest tools, it is possible to optimize routes with more than 50 locations to visit.

5. Conclusions

One of the main conclusions that can be drawn from experimenting with the described TSP, is that optimizing problems are rarely straightforward. Often, companies who provide cargo route optimization services market themselves as capable of solving any problem in a matter of seconds. However, even if their methods and models are state-of-the-art, optimization problems are difficult to solve as the “correct” result or optimized route cannot be quickly checked. Therefore, it is advisable to use multiple different approaches and compare the results. Furthermore, this implies that companies that engage in optimizing activities require skilled personnel to run these activities. It is not enough that a ready-made model is run with the problem specific inputs, but those who run it must be able to analyze the results and have required expertise to come up with possible ways to improve the optimization results. Often in industry starting ground of e.g., used

![Fig. 6. Another screenshot of Excel Solver add-on solving the same problem.](image-url)
distribution routes are rather farfetched from optimal solution that near optimal solutions are also workable and much valued. This article shows that optimization which is accurate and fast enough can be reached with openly available tools and accessible to companies with less resources to tribute for optimization activities.

Regarding the educational aspects of this work, it was discovered that the semi-open setting of the course, where students were given the tools, but not the answers, was successful in engaging students to optimize problems (analysis itself is more valuable than an exact and correct answer). Similarly, as [6], a soft introduction to the quantitative models helped the students to perform well in the course. The students that are already working in the logistics industry especially enjoyed the course (in student feedback evaluation course received grades around average of four as scale is from poor of one to excellent of five). Also, the playfully competitive setting helped to refine better solutions among the students. Moreover, real-life, substantial optimization problem helped the students to grasp the subject of the course, as was the case with [5].

In future, we would like to expand TSP optimization problem with different factors, such as cargo capacities, time windows, multiple vehicle fleets, and sustainability related challenges. It would be beneficial to showcase these more difficult optimization problems are solvable with rather simple tools. We recognize the notion of [6] and develop course materials in a way that there are more easy-to-use and

Table 1
Results over time with the algorithm using Google OR-Tools libraries.

<table>
<thead>
<tr>
<th>Runtime (milliseconds)</th>
<th>Optimized route length (km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25095</td>
</tr>
<tr>
<td>100</td>
<td>18978</td>
</tr>
<tr>
<td>200</td>
<td>18978</td>
</tr>
<tr>
<td>300</td>
<td>18978</td>
</tr>
<tr>
<td>400</td>
<td>18978</td>
</tr>
<tr>
<td>500</td>
<td>18978</td>
</tr>
<tr>
<td>600</td>
<td>18978</td>
</tr>
<tr>
<td>700</td>
<td>18978</td>
</tr>
<tr>
<td>800</td>
<td>18978</td>
</tr>
<tr>
<td>900</td>
<td>18978</td>
</tr>
<tr>
<td>1000</td>
<td>18978</td>
</tr>
<tr>
<td>1100</td>
<td>18978</td>
</tr>
<tr>
<td>1200</td>
<td>18662</td>
</tr>
<tr>
<td>1300</td>
<td>18662</td>
</tr>
<tr>
<td>1400</td>
<td>18662</td>
</tr>
<tr>
<td>1500</td>
<td>18662</td>
</tr>
<tr>
<td>1600</td>
<td>18662</td>
</tr>
<tr>
<td>1700</td>
<td>18662</td>
</tr>
<tr>
<td>1800</td>
<td>18662</td>
</tr>
<tr>
<td>1900</td>
<td>18615</td>
</tr>
<tr>
<td>2000</td>
<td>18610</td>
</tr>
</tbody>
</table>
easy-to-implement basic tools for the students, so that they might focus more on the optimizing rather than technical knowledge in software or programming languages. This is supported by the increase of additional computing power (through graphic cards) and development of Artificial Intelligence (AI), which are together bringing more advanced solutions for the masses in the forthcoming decades. As AI is already able to make programming work for humans, then modelling and interpretation of the solution is becoming increasingly more critical.

CRediT authorship contribution statement

Lähdeaho Oskari: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. Hilmola Olli-Pekka: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Supervision, Validation, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Python Script used in this study

```python
from ortools.constraint_solver import routing_enums_pb2.
from ortools.constraint_solver import pywrapcp.

def create_data_model():
    data = {}.
    data["distance_matrix"] = [.
        "INSERT DISTANCE MATRIX HERE".
    ].
    data["num_vehicles"] = 1.
    data["depot"] = 0.
    return data.

def print_solution(manager, routing, solution):
    print(f"Objective: {solution.ObjectiveValue()} miles").
    index = routing.Start(0).
    plan_output = "Route for vehicle 0:

    while not routing.IsEnd(index):
        previous_index = index.
        index = solution.Value(routing.NextVar(index)).
        plan_output += f " {manager.IndexToNode(index)} ->".
        plan_output += route_distance
        plan_output += f " Route distance: {route_distance} miles\n".
        print(plan_output).
        plan_output += f " Route distance: {route_distance} miles\n".
        return data = create_data_model().
        manager = pywrapcp.RoutingIndexManager(.
            len(data["distance_matrix"]), data["num_vehicles"], data["depot"]).
        routing = pywrapcp.RoutingModel(manager).
        def distance_callback(from_index, to_index):
            from_node = manager.IndexToNode(from_index).
            to_node = manager.IndexToNode(to_index).
            return data["distance_matrix"][from_node][to_node].
        transit_callback_index = routing.RegisterTransitCallback(distance_callback).
        routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index).
        search_parameters = pywrapcp.DefaultRoutingSearchParameters().
        search_parameters.time_limit.seconds = 500.
        search_parameters.log_search = True.
        solution = routing.SolveWithParameters(search_parameters).
        if solution:
            print_solution(manager, routing, solution).
            main:
        def main():
            data = create_data_model().
            manager = pywrapcp.RoutingIndexManager(
                len(data["distance_matrix"]), data["num_vehicles"], data["depot"]).
            routing = pywrapcp.RoutingModel(manager).
            def distance_callback(from_index, to_index):
                from_node = manager.IndexToNode(from_index).
                to_node = manager.IndexToNode(to_index).
                return data["distance_matrix"][from_node][to_node].
            transit_callback_index = routing.RegisterTransitCallback(distance_callback).
            routing.SetArcCostEvaluatorOfAllVehicles(transit_callback_index).
            search_parameters = pywrapcp.DefaultRoutingSearchParameters().
            search_parameters.local_search_metaheuristic = (.r
            routing_enums_pb2.GUIDED_LOCAL_SEARCH). search_parameters.time_limit.seconds = 500.
            search_parameters.log_search = True.
            solution = routing.SolveWithParameters(search_parameters).
            if solution:
                print_solution(manager, routing, solution).
                main:

References

```